**Научно-исследовательская работа**

на тему: **Обнаружение паттернов в моделях процессов с помощью свёрточных нейронных сетей**

**Ключевые слова:** Паттерны, process mining, свёрточные нейронные сети, модель процесса, Image-data engineering

2023

**Аннотация**

Моделирование применяется для исследования сложных процессов, обнаружения в них ошибок и их улучшения. Среди подходов для моделирования процессов можно выделить методики process mining. Алгоритмы process mining синтезируют модели процессов на основе журналов событий путём обнаружения закономерностей в виде паттернов. Существуют чисто алгоритмические подходы к синтезу, а также подходы, основанные на статистических закономерностях. В данном проекте исследуются статистические подходы к выделению паттернов моделей процессов с помощью свёрточных нейронных сетей, специфической особенностью которых является способность к последовательному выделению признаков в исследуемом наборе данных.

**Классификация ACM CSS**: Computing methodologies ~ Machine learning ~ Machine learning approaches ~ Neural Networks

## Оглавление

## Введение

В современном мире вокруг человека происходит масса процессов, многие из которых являются цифровыми. Процессы пронизывают все сферы нашей жизни, от личной коммуникации и социальных сетей до сложных корпоративных систем и научных исследований. Мы живем в эпоху, когда информация стала одним из самых ценных ресурсов. Подобное многообразие процессов требует эффективных инструментов для управления ими. Здесь на помощь человеку приходят *процессо-ориентированные информационные системы* (*ПОИС*). Эти системы предназначены для моделирования, управления, мониторинга и оптимизации процессов, делая их более эффективными и прозрачными.

Обнаружение, отслеживание и улучшение процессов возможно выполнять автоматизировано, используя в качестве исходных данных журналы событий, которые в большом количестве записываются информационными системами в процессе их функционирования, — такая идея лежит в основе семейства подходов под общим названием *process mining*. Process mining — относительно молодая исследовательская дисциплина, объединяющая традиционный анализ процессов на основе моделей (*process science*) и интеллектуальный анализ данных (*data science*).

Дисциплина process mining включает в себя четыре основных направления:

- *Обнаружение (синтез) процессов* (discovery) — подход к автоматизированному конструированию моделей процессов, которые, как правило, имеют графическую нотацию, а также интерпретацию, описывающую определенные аспекты процесса, например по управлению, используемым ресурсам или даже их комбинации. Это позволяет получить объективную картину реального процесса, независимо от того, как он описан в документах или регламентах.
- *Проверка соответствия модели процесса и его реального поведения* (conformance checking) — процедура проверки того, соответствует ли фактическое выполнение бизнес-процесса, записанное в журнале событий, модели и наоборот.
- *Улучшение процессов* (enhancement) — задача выявления узких мест и неэффективных действий, которые могут быть устранены для повышения продуктивности процессов и операций.
- *Расширение модели процесса* (extension) — добавление новых элементов в модель процесса, таких как ограничения, атрибуты или метрики.

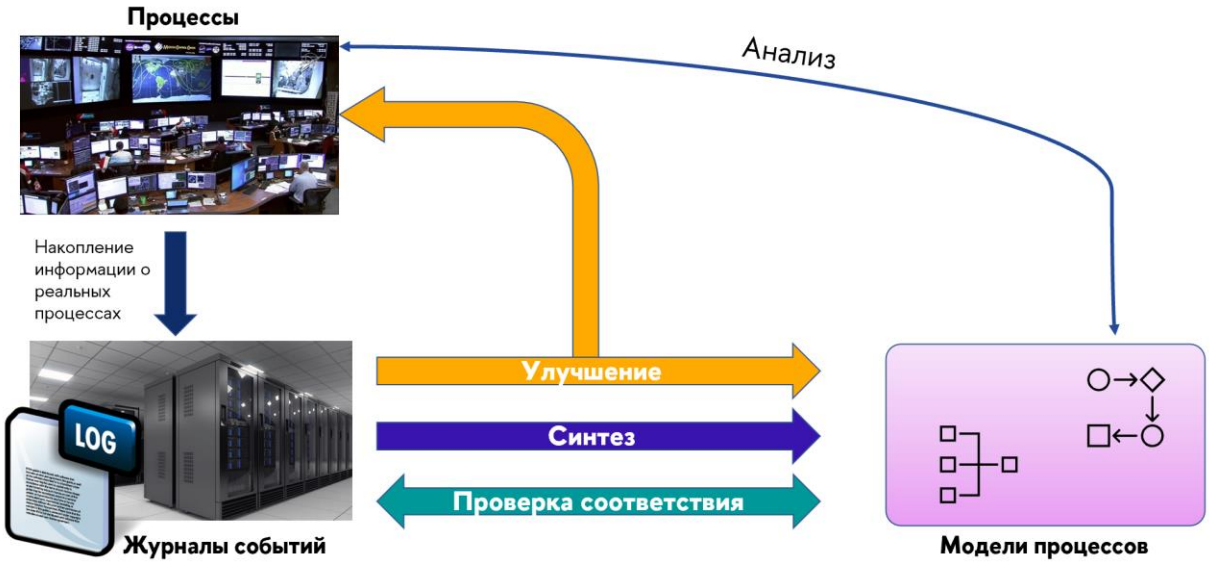



Рисунок 1 - Позиционирование основных направлений process mining.

Центральным элементом для process mining является *активность* (activity) — это конкретная задача или действие, которое выполняется в рамках некоторого бизнес-процесса. Повторяющиеся регламентированные исполнения конкретного бизнес-процесса называются его *экземпляром* (instance). Реализацией определённой активности в рамках конкретного экземпляра бизнес-процесса является *событие (event).* Событие можно рассматривать как фундаментальный строительный блок процесса, представляющий собой единицу работы, выполняемую человеком — участником процесса либо некоторой системой или ее компонентом (актором). Каждое действие в бизнес-процессе обычно определяется своим именем, уникальным идентификатором и набором атрибутов. Теперь, пусть $A$ — множество активностей. *Трасса* (*событий*) представляет собой последовательность $\sigma = \langle a_1, a_1, \dots , a_i \dots , a_n \rangle \in A^+$. С помощью символа $a_i$ мы обозначаем $i-$*ый* элемент (событие) трассы. *Журналом событий* (логом событий) является $L \in B(A^+)$, $|L|$ — размер журнала (лога), равный количеству содержащихся в нём трасс. Таким образом, журнал событий является мультимножеством над множеством трасс событий. Здесь и далее мы подразумеваем, что журнал событий содержит по крайней мере одну трассу.

$$L = [\langle a, c, d, f \rangle, \langle a, d, c, f \rangle, \langle a, b, f \rangle, \langle a, e, f \rangle] \qquad (1)$$

Здесь $L$ - пример журнала событий, содержащего 6 активностей, 4 трассы и 14 событий, где активности закодированы буквами $a$, $b$, $c$, $d$, $e$, $f$, трассы заключены в треугольные скобки, а события представлены теми же буквами внутри трасс, но при этом неявно, помимо отсылки к исполняемой активности, содержат в качестве атрибутов порядок появления в конкретной трассе.

Поэтому событие c из трассы $\langle a, c, d, f \rangle$ можно более полно представить в виде тройки $(c, 1, 2)$, где c — это идентификатор активности, 1 — это номер трассы по порядку, начиная с 1, 2 — это номер события внутри трассы 1. Событие a из трассы $\langle a, d, c, f \rangle$ аналогичным образом можно представить в виде тройки $(c, 2, 3)$.

Одной из задач, для которой применение методов process mining может дать хорошие результат, является обнаружение поведенческих паттернов (или шаблонов поведения) в бизнес-процессах. Именно это направление исследуется в рамках данной работы, которая находится на стыке двух исследовательских направлений process mining — синтеза модели и проверки соответствия модели и реального поведения. Обнаружение паттернов относиться к проверкам взаимодействий в рамках некоторого процесса, а синтез модели обнаруженного паттерна, относиться к задаче discovery. Однако значительные масштабы реальных процессов ведут к усложнению задачи обнаружения паттернов в бизнес-процессах. Обнаружение паттернов путём синтеза модели процесса может быть затратным по времени и ресурсам, при этом она может содержать неточности и не в полной мере отражать действительную организацию процесса.

Машинное обучение стало ключевым инструментом для решения множества прикладных задач, в частности, для выявления паттернов и закономерностей в различных данных. Глубокое обучение — тип машинного обучения, основанный на применении искусственных нейронных сетей, которые путём нескольких уровней обработки и повышения абстракций производят извлечение признаков высокого уровня из данных. Отличие от традиционных методов process mining, заключается в том, что нейронные сети обеспечивают высокую степень автоматизации и могут эффективно масштабироваться для анализа больших и сложных наборов данных. Это делает их гибким и мощным инструментом для выявления паттернов в различных доменах и сценариях.

**Целью** данной работы является разработка метода выделения поведенческих паттернов в моделях процессов без синтеза самой модели процесса (то есть непосредственно из исходных данных, коими являются журналы событий) с применением методов машинного обучения. Прикладной составляющей работы является исследование журнала событий на предмет заранее определённых паттернов (взаимодействий) с целью проверки процесса на отсутствие недопустимых поведений (переходов) или на наличие и состояние всех необходимых переходов.

Для достижения данной цели необходимо решить следующие задачи:

- исследовать существующие технологии, которые применялись к направлениям process mining;
- определить тип нейронной сети для извлечения поведенческих паттернов;
- разработать архитектуру нейронной сети;
- разработать способ представления журналов *событий для понимания нейронной сетью*;
- сформировать обучающий набор данных;
- произвести обучение и тестирование нейронной сети;

Данная работа организована следующим образом. Глава 1 даёт обзор источников по теме process mining и смежных работ по использованию методов машинного обучения в данной тематике. В главе 2 представлено описание конфигурации нейронной сети. Глава 3 описывает подход по кодированию трасс в форме матриц (изображений). Глава 4 предоставляет поэтапный алгоритм обнаружения паттернов, а глава 5 — особенности его реализации. В главе 6 приводится описание произведённых экспериментов. В заключении подводятся результат работы и предлагается дальнейшие направления исследования.

## 1. Существующие решения и технологии

Основным способом обнаружения паттернов в бизнес-процессах является построение модели процесса (как правило, графовой, с семантикой параллелизма или без неё), и впоследствии её дальнейшее исследование, включающее обработку и визуализацию. Наиболее популярными моделями являются сети Петри, BPMN нотации, process trees и Workflow сети [1].

Для синтеза моделей процессов разработано большое число алгоритмов, имеющих в основе весьма разные принципы функционирования. Классическим является *алгоритмический подход*, который лежит в основе детерминированных алгоритмов, которые для одного и того же набор входных данных (журнала событий) при одном наборе настроек дают одинаковый результат. К примерам алгоритмического подхода можно отнести алгоритмы Alpha Miner [2], Inductive Miner и Heuristic Miner [1]. Существуют также альтернативные подходы по синтезу моделей процессов. Они основываются на методах машинного обучения. Так, например, Sommers и др. в [3] используют *графовые нейронные сети* для построения сети Петри.

Предложенный подход основа на *обучении с учителем* (supervised learning), одной из основных методик машинного обучения. Авторами разработана модель (нейронная сеть), которая была натренирована на синтетически сгенерированных журналах событий и соответствующих им сетях Петри. В результате модель способна транслировать реальные журналы (логи) в сети Петри. В исследовании [7] применяют *рекуррентные нейронные сети* для создания *системы переходов* (transition systems - TS), ещё одного из способов представления модели процесса. Авторы представляют метод, при котором рекуррентная сеть обучается на основе журналов событий, а внутреннее состояние сети используется для формирования желаемой TS, отражающей поведение, зафиксированное в логах. Входными параметрами в данной конфигурации сети является последовательность событий, закодированных числами — трасса.

В работах [4], [5] авторы описывают подходы по определению возможного следующего события в трассах в поданном на вход журнале событий путём выделения *префикс-трасс*, кодирования их специальным образом в виде монохромных изображения и обучения *свёрточной нейронной сети*. Здесь префикс-трасса — это подпоследовательность трассы, формирующаяся с её начала. Свёрточная нейронная сеть обучается на сгенерированных изображениях, и после обучения она способна предсказывать возможное следующее событие в трассах нового журнала, который не участвовал в обучении нейронной сети.

В статье [6] авторы предлагают подход, основанный на представлении паттернов и всего журнала событий в виде *BP-графов* (Business Process Graphs). Данные графы подаются на вход алгоритму, который производит их анализ посредством сравнения вершин графов и их атрибутов. В процессе анализа для вершин графа журнала событий устанавливаются значения сходства с паттерном (label similarity values). Все вершины графа, значения сходства которых выше порогового значения, считаются совпадающими с паттерном. Важно, что паттерном здесь называется состояние вершины, то есть набор её атрибутов, а не структурная часть процесса, состоящая из вершин и переходов между ними.

Работа [8] относится к *проверке соответствия* — направлению process mining. Предложена структура, использующая две рекуррентные нейронные сети, которые проходят трассу из журнала событий в обе стороны, и каждая нейронная сеть, пройдя некоторое количество шагов, предсказывает

возможное следующее событие. После производится сравнение двух полученных трасс.

Алгоритмические подходы, отлично зарекомендовавшие себя в сообществе исследователей process mining, методы на основе нейронных сетей, а также работы по предсказанию следующего события в рамках процесса позволяют с определённой степенью точности представить весь процесс в рамках одной модели, например сети Петри. Однако они не позволяют напрямую определять структурные составляющие процесса, а именно те высокоуровневые конструкции, из которых состоит весь процесс, и которые мы в данной работе называем паттернами. Данным исследованием мы предлагаем новый метод для решения данной проблемы.

## 2. Конфигурация нейронной сети

Большой объём журналов событий и необходимость производить предварительный анализ входных данных для понимания структуры взаимодействующих элементов процесса являются большими ограничениями быстрого и эффективного обнаружения паттернов методами process mining. В рамках данной работы мы выдвигаем гипотезу, что подходы из области машинного обучения позволяют снять подобные ограничения. Среди множества методов машинного обучения, свёрточные нейронные сети являются одним из наиболее эффективных видов моделей для извлечения паттернов в данных. Они были успешно применены в различных задачах, включая распознавание изображений, компьютерное зрение, естественный язык и анализ временных рядов [13, 14, 15, 16].

Свёрточные нейронные сети обладают рядом преимуществ, которые делают их особенно подходящими для решения задачи обнаружения паттернов в моделях процессов. Во-первых, они способны обнаруживать локальные закономерности в данных, что важно для задач, связанных с выявлением повторяющихся элементов. Во-вторых, они обладают способностью к масштабированию, что позволяет им обрабатывать данные больших размеров.

Особенностями свёрточных нейронных сетей является использование трёх основных принципов: *свёртки*, *субдискретизации* (*пулинга*) и *общих весов* [4]. Основными типами слоёв в свёрточных нейронных сетях являются — свёрточные слои (*convolutional layer*), пулинговые слои (*pooling layer*) и полносвязные слои (*fully-connected layer*) [9]. Свёрткой является операция над

матрицами $M_{n\times k}$ и $K_{m\times l}$, при соблюдении условия $n \geq m$ и $k \geq l$. Матрица $M$ является входными данными для очередного свёрточного слоя нейронной сети, а матрица $K$ является ядром (фильтром), которым и производиться свёртка изображения. Результатом данной операции над матрицами $M$ и $K$ является матрица $C$, которая будет иметь размер $(n - m + 1) \times (k - l + 1)$. Элементы матрицы $C$ вычисляются как скалярное произведение матрицы $K$ и подматрицы $M$ того же размера. Таким образом, $C_{i,j} = \sum_{u=0}^{m-1} \sum_{v=0}^{l-1} M_{i+u,j+v} K_{u,v}$. У свёрточного слоя есть гиперпараметр, называемый *сдвигом* (stride). Он определяет на сколько элементов матрицы $M$ будет производиться сдвиг фильтра $K$. Процесс свёртки матрицы $5 \times 5$ с фильтром $3 \times 3$ со сдвигом 1 показан на Рисунке 2 [9], зелёным цветом представлена результирующая матрица $C$.

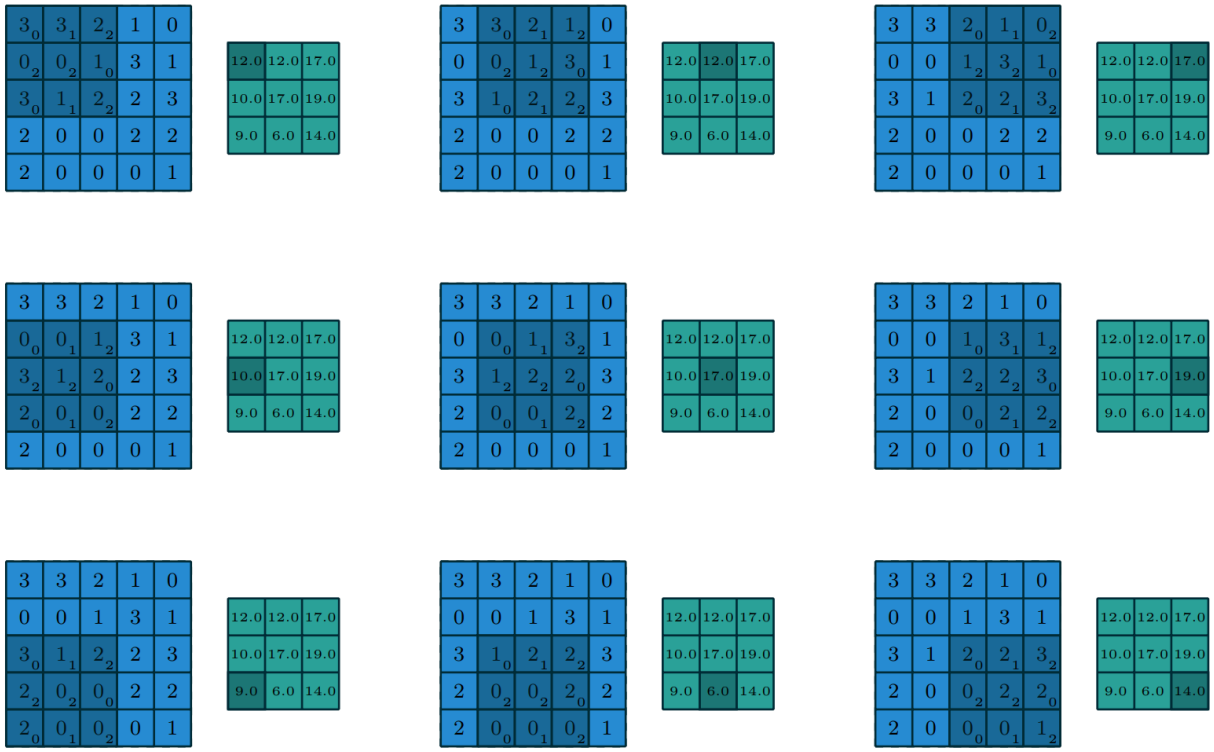


Рисунок 2 — Процесс свёртки матрицы.

Выбор матрицы фильтра в свёрточной нейронной зависит от признака, который необходимо определить во входной матрице. Обычно используется несколько фильтров для выявления разных признаков, их количество — еще один гиперпараметр сети и на практике определяется исключительно экспериментально. Фильтры являются обучаемыми параметрами сети, и сеть сама корректирует их в процессе обучения. Следует отметить, что после каждой свёртки применяется функция активации *ReLu* $f(x) = max(0, x)$, которая впервые была представлена в [11] и является одной из наиболее часто

используемых в машинном обучении для внедрения нелинейности, улучшая тем самым способность модели к обучению. Далее применяется один из трёх видов пуллинга — слоя, который сокращает пространственные размеры полученных после свёртки матриц для ускорения вычислений без потери существующей информации. Кроме того, может применяться слой *нормализации* (batch normalization), который оптимизирует процесс, ускоряя сходимость обучения. Его идея заключается в следующем — преобразовывать выходы слоёв так, чтобы они гарантированно имели фиксированное распределение. Сначала происходит подсчёт среднего и дисперсии по всем столбцам входной матрицы слоя нормализации, а потом — центрирование и нормирование её элементов. Тогда по всем столбцам будет нулевое среднее и единичная дисперсия. Однако в таком случае возникает проблема, что после применения нелинейности $ReLu$ все отрицательные значения матрицы будут занулены, поэтому их умножают на новую дисперсию $\gamma$ и прибавляют новое среднее $\beta$, $\gamma$ и $\beta$ — обучаемые параметры нейронной сети, которые адаптируются в процессе алгоритма обучения.

После всех этапов свёртки и пулинга следует процесс выравнивания (flattening), где многомерные данные преобразуются в одномерный вектор, который затем подается на вход полносвязным слоям для выполнения классификации. Полносвязные слой можно представлять как функцию, которая на вход принимает $n$ чисел, а на выходе предоставляет $m$ чисел. Пусть $(x_1, \dots, x_n)$ — вектор входных чисел, а $(z_1, \dots, z_n)$ — вектор выходных, в полносвязном слое каждый выход является линейной моделью над входами $z_j = \sum_{i=1}^{n} w_{ji}x_i + b_j$, где $w_{ji}$ — веса модели и $b_j$ — свободный коэффициент, данные параметры определяются моделью в процессе обучения. Получившуюся линейную модель $z_j$ в сочетании с функцией активации, например $ReLu$, называют *нейроном*. Здесь стоит упомянуть о слое *dropout*, который используется для предотвращения переобучения путем случайного исключения нейронов во время обучения.

В задачах классификации к последнему полносвязному слою применяется функция активации $softmax$ [12], которая возвращает $n$-мерный вектор, где $i$-ый элемент является вероятностью принадлежности поданного на вход изображения к одному из $n$ классов. Класс с наибольшей вероятностью становиться предсказанным классом изображения.

Важно отметить, что обучение всей сети осуществляется с использованием метода *обратного распространения ошибки* (backpropagation) *и градиентного*

*спуска* (gradient descent). Обратное распространение ошибки — это метод, используемый для обновления весов в нейронной сети путем распространения ошибки обратно от выходов к входам. Под ошибкой подразумевается заранее определённая функция, которая будет показывать, на сколько модель далека в своём предсказании от правильного ответа. После прохождения вперед (прямой проход данных через сеть и получение предсказания), рассчитывается ошибка, которая затем обратно распространяется по сети, и градиенты ошибки используются для корректировки весов в направлении, которое минимизирует ошибку. Градиентный спуск — это оптимизационный алгоритм, используемый вместе с обратным распространением, чтобы "спускаться" по поверхности ошибки, настраивая веса таким образом, чтобы достичь наименьшей возможной ошибки или потерь. Конечно, существуют и другие алгоритмы обучения — эволюционные алгоритмы, алгоритм имитации отжига, градиентный бустинг, случайный поиск, однако градиентный спуск остаётся популярным из-за его эффективности в смысле масштабируемости и обобщающей способности при работе с большими нейронными сетями.

Следует также упомянуть о важности *коэффициента скорости обучения* (learning rate) в процессе градиентного спуска. Этот параметр контролирует величину шага обновления весов во время обучения и имеет решающее значение для сходимости модели. Слишком большой коэффициент может привести к нестабильности и колебаниям, в то время как слишком маленький замедляет процесс обучения и может застрять в локальных минимумах. Эффективное значение коэффициента скорости обучения часто определяется экспериментальным путем, и его настройка является важным аспектом обучения нейронных сетей.

В данной работе применяется вариант архитектуры свёрточной нейронной сети, который был предложен в нейронной сети VGG [14], он показан на Рисунке 3. Слои, применяемые в нейронной сети, изображены прямоугольниками. В архитектуре присутствует три блока свёртки — $conv1, conv2, conv3$, а также два полносвязных слоя. Все свёртки производиться с ядром одного размера $3 \times 3$, подобный тип ядра показал свою эффективность по сокращению обучаемых параметров в архитектуре VGG [14]. Первый свёрточный слой $conv1$, он показан жёлтым цветом, выполняет роль входного, как представлено на Рисунке 3 ему подаётся изображение размера $256 \times 256$ пикселей, с одним цветовым каналом. В результате свёртки 16 различными ядрами (фильтрами) $3 \times 3$ мы получаем 16 матриц размер которых можно рассчитать по ранее указанной формуле $(n - m + 1) \times$

$(k - l + 1)$, где $n \times k = 256 \times 256$ — размеры входной матрицы, а $m \times l = 3 \times 3$ — размеры фильтра, тогда получаем матрицы размера $254 \times 254$. После применения нелинейности к свёрткам происходит нормализация матриц, на Рисунке 3 изображена сиреневым цветом, она позволяет ускорять вычисления и избегать переобучение нейронной сети. Данные операции никак не влияют на размер матриц. В конце блока $conv1$ происходит операция макс-пулинга, для эффективного выделения наиболее ярких признаков. Операция в данной архитектуре всегда происходит с размерностью $2 \times 2$, то есть матрица размера $254 \times 254$ делиться на подматрицы размера $2 \times 2$ и в данной подматрице выбирается максимальный элемент, остальные отбрасываются. Поэтому размер матрицы уменьшается в два раза. Выбор параметра $2 \times 2$ в макс-пулинге эффективно уменьшает размер входных данных для сокращения количества параметров и снижает вычислительную нагрузку. Таким образом, после применения блока $conv1$ для изображения размером $256 \times 256$ пикселей мы получаем 16 матриц размера $127 \times 127$, именно они и идут на вход свёрточному слою блока $conv2$, а его результаты на вход блока $conv3$.
После трёх блоков свёртки мы получаем набор матриц, к которому применяем операцию преобразования к одномерному массиву (flattening), то есть все полученные матрицы приводятся к одному вектору. Данный вектор подаётся на вход первому полносвязному слою, который изображен на Рисунке 3 синим цветом, а его выходы применяются к последнему зелёному полносвязному слою с функцией активации $softmax$. На выходе последнего полносвязного слоя мы получаем $n$ −мерный вектор вероятностей принадлежности входного изображения к одному из $n$ классов. Данный вектор является результатом работы свёрточной нейронной сети.

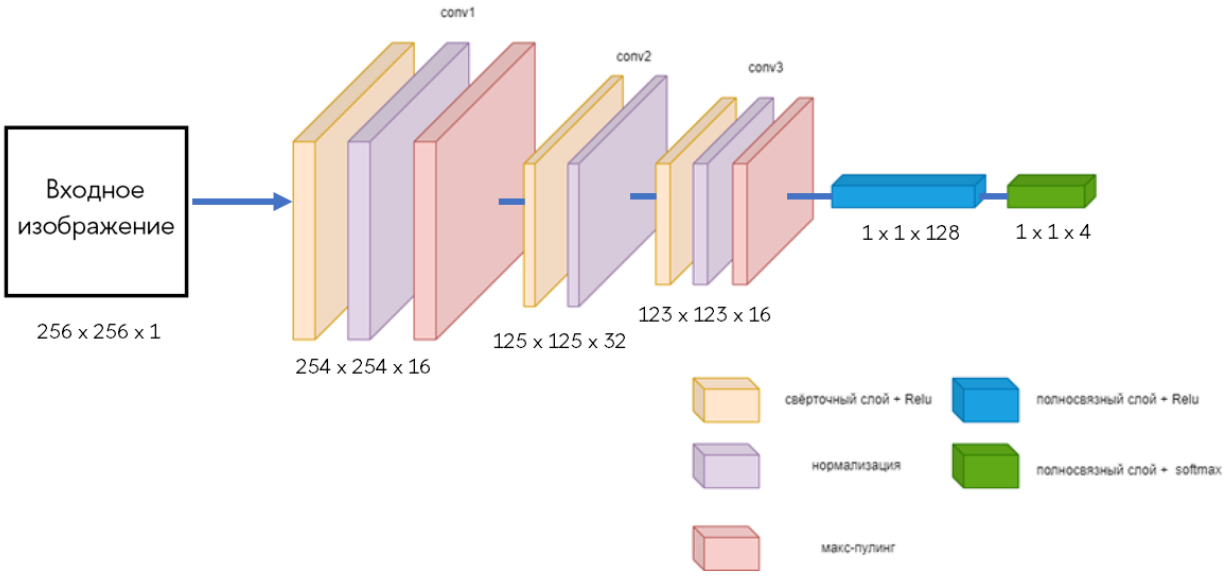



Рисунок 3 — Архитектура свёрточной нейронной сети.

## 3. Кодирование журнала событий

Традиционное использование свёрточных нейронных сетей — классификация изображений, представлением которых является матрица чисел, соответствующих определённому цветовому каналу. Для задачи определения паттернов в журнале событий с помощью нейронной сети необходимо использовать метод для представления журнала или его частей как изображения. В процессе работы над проектом проведено исследование по возможным способам представления журналов событий в виде матриц и выделено 11 различных вариантов. Итоговый выбор был сделан в пользу метода Image data engineering, представленного ранее в нескольких работах [4], [5]. Он основывается на том, что журнал событий несёт информацию о каждом событии, произошедшем в рамках экземпляра бизнес-процесса, с определённой продолжительностью во времени. Таким образом, каждое событие обладает двумя необходимыми свойствами: соответствует некоторой активности из множества активностей $A$ — $act_i$ и имеет временную метку $ts_i$, являющуюся датой и временем начала выполнения события. Рассматриваемое множество $A$ — конечно и определяется набором различных событий, которые могут возникнуть в соответствии с моделью рассматриваемого бизнес-процесса.

| ID Трассы | Активность | Время |
|---|---|---|
| 1 | Регистрация запроса (РЗ) | 2022-11-30 10:06 |
| 1 | Проверка билета (ПБ) | 2022-11-30 10:30 |
| 1 | Тщательное изучение (ТИ) | 2022-11-30 11:02 |
| 1 | Проверка билета (ПБ) | 2022-12-01 15:32 |
| 1 | Тщательное изучение (ТИ) | 2022-12-01 16:20 |
| 1 | Принятие решения (ПР) | 2022-12-01 16:30 |
| 1 | Выплата компенсации (ВК) | 2022-12-03 12:25 |
| 2 | Регистрация запроса (РЗ) | 2022-12-10 13:15 |
| 2 | Проверка билета (ПБ) | 2022-12-11 16:20 |
| 2 | Принятие решения (ПР) | 2022-12-11 17:40 |
| 2 | Отклонение запроса (ОЗ) | 2022-12-12 14:20 |
| 3 | … | … |

Таблица 1— пример фрагмента журнала событий. Множество активностей $A$ состоит из 6 различных активностей. Каждое событие в журнале соответствует активности из множества $A$ и имеет временную метку

Для осуществления кодирования журнала в подходе Image Data Engineering предлагается использовать представление его составных частей — трасс в виде матриц, которые впоследствии станут изображениями. Перед описанием процесса кодирования трасс необходимо ввести понятие *префикс-трассы*. Префикс-трассой ($PT_k$) мы назовём первые $k$ событий трассы $\sigma$, из трассы длиной $l = |\sigma|$ можно получить $l$ префикс-трасс. Префикс-трасса $PT_k$ строиться для события $a_k$, то есть это будет последовательность первых $k$ событий, включая $a_k$, $PT_k = \langle a_1, \ldots, a_k \rangle$. Рассмотрим журнал событий, состоящий из активностей — РЗ, ПБ, ТИ, ПР, ВК, ОЗ и имеющий значения ID трассы и время начала исполнения (Таблица 1). По данному журналу событий построим набор префикс-трасс по трассе с ID = 1 (Таблица 2).

| **ID Трассы** | **ID Префикс-трассы** | **Префикс-трасса** |
|---|---|---|
| 1 | 1 | (РЗ; 2022-11-30 10:06) |
| 1 | 2 | (РЗ; 2022-11-30 10:06), (ПБ; 2022-11-30 10:30) |
| 1 | 3 | (РЗ; 2022-11-30 10:06), (ПБ; 2022-11-30 10:30), (ТИ; 2022-11-30 11:02) |
| 1 | 4 | (РЗ; 2022-11-30 10:06), (ПБ; 2022-11-30 10:30), (ТИ; 2022-11-30 11:02),<br>(ПБ; 2022-12-01 15:32) |
| 1 | 5 | (РЗ; 2022-11-30 10:06), (ПБ; 2022-11-30 10:30), (ТЗ; 2022-11-30 11:02),<br>(ПБ; 2022-12-01 15:32), (ТИ; 2022-12-01 16:20) |
| 1 | 6 | (РЗ; 2022-11-30 10:06), (ПБ; 2022-11-30 10:30), (ТИ; 2022-11-30 11:02), (ПБ; 2022-12-01 15:32), (ТИ; 2022-12-01 16:20), (ПР; 2022-12-01 16:30) |
| 1 | 7 | (РЗ; 2022-11-30 10:06), (ПБ; 2022-11-30 10:30), (ТИ; 2022-11-30 11:02),<br>(ПБ; 2022-12-01 15:32), (ТИ; 2022-12-01 16:20), (ПР; 2022-12-01 16:30),<br>(ВК; 2022-12-03 12:25) |

Таблица 2— Набор префикс-трасс, построенный по трассе из журнала на Рисунке 4

Таблица 2 содержит число строк по числу событий, представленный в данной трассе, что равно количеству префикс-трасс. В ней также есть 3 столбца, *ID Трассы* и *ID Префикс-трассы* отражают идентификатор элемента, трассы или префикс-трассы, для которого строиться набор, а столбец *Префикс-трасса*

является упорядоченной последовательностью $k$ первых элементов трассы, что и является выделенной префикс-трассой.

По полученному набору префикс-трасс для каждой трассы $T_i$ , $i$ изменяется от 1 до числа трасс в журнале событий, строится *отдельная* матрица $M_i$, которая в дальнейшем будет перекодирована в изображение. Матрица $M_i$ имеет размер $l \times h$, где $l$ — длина трассы $T_i$, для которой получен набор префикс-трасс, это число также отражает число моментов времени, в которые происходили события в рамках конкретной трассы; $h$ — количество активностей в множестве всех допустимых активностей — $A$. За каждым столбцом закреплена определённая активность $act_i$ — порядок их следования должен быть строго определён и быть одинаковым для всех изображений, строящихся как для обучения свёрточной нейронной сети по набору паттернов, так и при переводе трасс из журнала событий. Таким образом, каждое $(i, j)$-ое значение в матрице отражает количество раз, которое событие, соответствующее активности $act_j$ из множества $A$, появилось на момент времени $i$ в трассе $T_i$. Пример подобной матрицы для префикс-трасс из Таблицы 2 представлен в Таблице 3.

| **Время активность** | **РЗ** | **ПБ** | **ТИ** | **ПР** | **ВК** |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 | 0 |
| 3 | 1 | 1 | 1 | 0 | 0 |
| 4 | 1 | 2 | 1 | 0 | 0 |
| 5 | 1 | 2 | 2 | 0 | 0 |
| 6 | 1 | 2 | 2 | 1 | 0 |
| 7 | 1 | 2 | 2 | 1 | 1 |

Таблица 3 — Матрица префикс-трассы.

Таблица 3 содержит число строк по количеству моментов времени, в которые в трассе появлялось новое событие, что равно количеству событий в трассе. Число столбцов соответствует количеству допустимых активностей в рамках экземпляра бизнес-процесса, то есть это мощность множества $A$. Так число 1 на пересечении строки 1 и столбца РЗ показывает, что в данный момент времени активность РЗ появлялась в трассе только 1 раз, а, например, число 2 на пересечении строки 5 и столбца ТИ показывает, что к тому моменту активность ТИ была представлена в трассе двумя событиями.

Построенную матрицу $M_i$ можно рассматривать как двумерное изображение $I_i$, интерпретируя значения в ячейках в виде интенсивности цветового оттенка серого (Рисунок 4). Так же, как и в матрице $M_i$, строки изображения $I_i$ отражают моменты времени в трассе, а столбцы — последовательность активностей из множества $A$. Изначально в изображении содержится $l * h$ пикселей, по количеству значений в матрице префикс-трасс $M_i$ трассы $T_i$. Каждый $(i, j)$ пиксель имеет значение соответсвующее значению матрицы (i, j), нормированное на отрезке $[0, 255]$. Таким образом, значение нуля в матрице соответствует нулевой интенсивности цвета, что обычно является чёрным цветом на изображении $I_i$, а максимальное значение (как значение 2 в Таблице 2) — соответственно белому цвету.

Технической особенностью данного подхода является то, что все изображения в обучаемом наборе должны быть одного размера. Однако в журнале событий трассы (соответствующие экземплярам процессов), как правило, имеют различную длину: даже один и тот же процесс в зависимости от условий протекания может включать разное число действий-активностей. Поэтому длины префикс-трасс также будут отличаться, то есть для каждой трассы матрица подобная матрице из Таблицы 3 будет иметь различное число столбцов. Вследствие чего и размеры изображений, являющихся 2D представлением префикс-трассы, будут различными. Для устранения этой проблемы матрицы коротких трасс дополняются строками нулей, в результате чего выравниваются размеры всех матриц $M_i$.

Рисунок 4 — Интерпретация матрицы префикс-трассы по Таблице 3 в виде двумерного изображения с палитрой оттенков серого

## 4. Обнаружение паттернов в моделях процессов

Входными данными для разрабатываемого алгоритма выделения паттернов является исходный журнал $L$, состоящий из ненулевого количества трасс. Выделим основные шаги этого алгоритма:

1) Для каждой трассы $\sigma \in L$ строится набор префикс-трасс.
2) Производится построение матриц на основе полученных префикс-трасс и подхода Image data engineering.
3) Каждая матрица переводится в двумерное изображение с палитрой оттенков серого.
4) Формируется набора изображений для обучения свёрточной нейронной сети.
5) Формируется интересующий набора паттернов перед обучением нейронной сети.
6) Создаётся синтетический журнал событий для каждого паттерна.
7) Трассы синтетических журналов перекодируются в изображения.
8) Нейронная сети обучается на полученном наборе изображений для задачи мультиклассификации, то есть классификации изображений более чем двух классов.

## 5. Программная реализация предложенного метода

В данном разделе описывается одна из возможных реализаций предложенного метода обнаружения паттернов на языке Python версии 3.11 с использованием библиотек данного языка. Для разработки алгоритма машинного обучения применяются инструменты Tensorflow [1] и Keras[2], которые являются наиболее популярными и широко распространёнными в данной области. TensorFlow — это открытая платформа для разработки и внедрения алгоритмов машинного обучения, созданная Google. Она позволяет создавать, обучать и развертывать различные модели глубокого обучения, в частности такие как сверточные нейронные сети. Keras — это высокоуровневый API для построения и обучения нейронных сетей, созданный на основе TensorFlow и в последствии интегрированный в него. Он предоставляет интерфейс для создания различного типа моделей машинного обучения, такие как последовательные модели, функциональные модели и модели с несколькими входами и выходами. Для обработки журналов событий, которые чаще всего хранятся либо в файлах формата CSV, либо XES используется специально разработанная Python библиотека для process mining — pm4py. Дополнительно

[1] https://www.tensorflow.org/
[2] https://keras.io/

также используются широко распространённые библиотеки *NumPy, Pandas, Matplotlib* для хранения, преобразования и визуализации данных.

Для подробного ознакомления с программным кодом можно воспользоваться jupyter notebook из репозитория на платформе github[3].

## 6. Экспериментальная проверка

Для тестированная предложенного в данной работе метода по определению паттернов были выбраны несколько примеров из классификации паттернов представленной Ramezani в [10], это: 1) *Between. After Before* 2) Response. Direct 3) *Existence. Activity universality* 4) *Bounded Existence of an Activity. Exactly k times*. Сети Петри [10], моделирующие данные паттерны представлены на Рисунке 5.

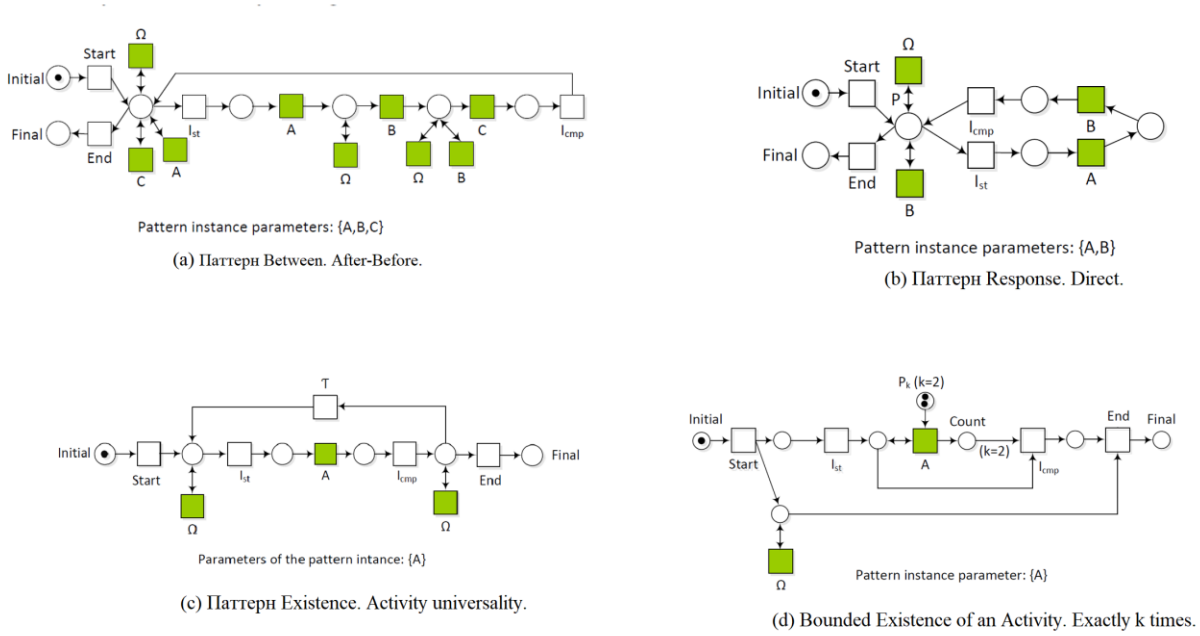
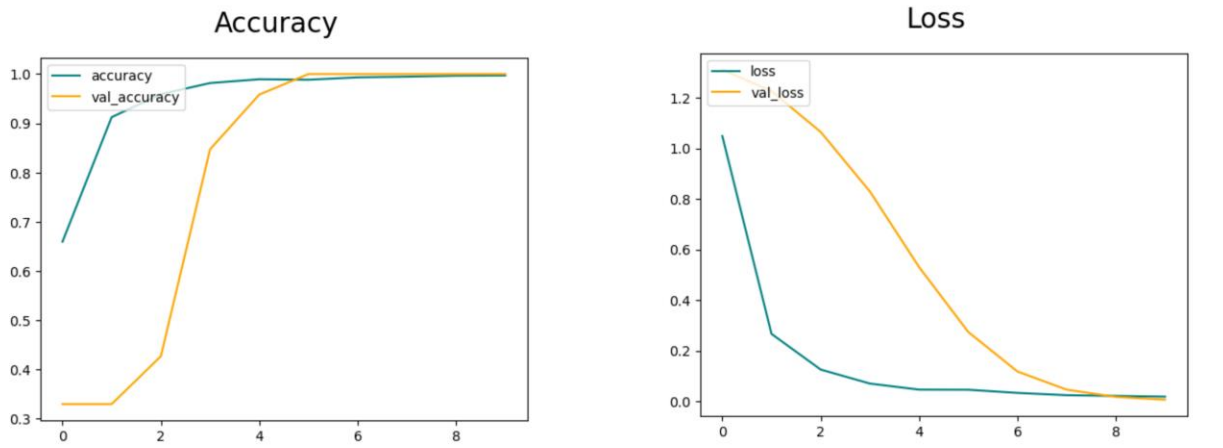



Рисунок 5 — Сети Петри паттернов.

Данные модели были построены в приложении WoPeD[4]. По ним были созданы журналы событий при помощи инструментов фреймворка для process mining ProM[5]. Для каждого паттерна было построено по 1000 трасс, которые были перекодированы в изображения методом Image data engineering, описанным ранее. Нейронная сеть была обучена на изображениях трасс, соответствующих каждой из категорий. Гиперпараметрами, которые возможно варьировать при обучении, являются количество эпох, и данной работе рассматривается сеть,

[3] github.com/ironSensei/PM_Pattern_Recognition
[4] https://woped.dhbw-karlsruhe.de/
[5] https://promtools.org/

которая обучалась в течение 10 эпох, и learning rate, выбранный на уровне $10^{-6}$. Результаты обучения свёрточной нейронной сети представлены на Рисунке 6, на нём отображены *accuracy*, который показывает долю правильно классифицированных примеров от общего числа примеров, а также *loss*, являющийся значением функции потерь. Данные метрики отражены как для этапа тестирования, так и для этапа валидации.




(a) Значение функции точности при обучении Свёрточной нейронной сети

(b) Значение функции потерь при обучении Свёрточной нейронной сети

Рисунок 6 — Показатели обучения нейронной сети.

После обучения нейронной сети, была проведена серия тестов по её работе. На вход ей подавался журнал событий, сгенерированный также при помощи инструментов WoPeD и ProM. Каждый журнал событий представляет собой комбинацию нескольких паттернов в рамках бизнес-процесса, отражаемого сетью Петри. Результаты работы нейронной сети представлены в Таблице 4.

| Комбинация паттернов | Верно определены | Ошибочно добавлены | Ошибочно пропущены |
|---|---|---|---|
| RD + BetwAB | 2 | 0 | 0 |
| RD + EAU | 1 | 0 | 1 |
| EAU + BetwAB | 2 | 1 | 0 |
| Bound3T + BetwAB | 1 | 1 | 1 |
| Bound3T + RD | 1 | 0 | 1 |
| Bound3T + EAU | 1 | 0 | 1 |

Таблица 4 — Результаты работы нейронной сети. RD – response direct pattern, BetwAB - Between. After Before, EAU - Existence. Activity universality, Bounded Existence of an Activity. Exactly 3 times.

Как видно из Таблицы 4, результаты работы свёрточной нейронной сети не являются абсолютно точными. Можно посчитать метрики precision, recall и F-1 меру.

$$precision = \frac{\text{Верно определены}}{\text{Верно определены} + \text{Ошибочно добавлены}}$$

Эта метрика показывает, какой процент объектов, определенных как положительные, действительно являются положительными.

$$recall = \frac{\text{Верно определены}}{\text{Верно определены} + \text{Ошибочно пропущены}}$$

Эта метрика показывает, какой процент реальных положительных объектов был правильно определен.

$$F1 = \frac{precision \times recall}{precision + recall}$$

F1-мера представляет собой гармоническое среднее между точностью и полнотой.

| **Комбинация паттернов** | **Точность** | **Полнота** | **F1-мера** |
|---|---|---|---|
| RD + BetwAB | 1.0 | 1.0 | 1.0 |
| RD + EAU | 1.0 | 0.5 | 0.67 |
| EAU + BetwAB | 0.67 | 1.0 | 0.8 |
| Bound3T + BetwAB | 0.5 | 0.5 | 0.5 |
| Bound3T + RD | 1.0 | 0.5 | 0.67 |
| Bound3T + EAU | 1.0 | 0.5 | 0.67 |

Таблица 5 — Показатели работы модели на основе трёх метрик

Модель, как правило, правильно определяет наличие одного из действительно представленных паттернов, но при этом может ошибочно показывать наличие непредставленного паттерна. Это можно объяснить наслоениями в исходных данных, то есть в рамках одной трассы может быть представлено несколько паттернов. В такой ситуации на изображении по такой трассе появляются наложения нескольких паттернов друг на друга, и свёрточной нейронной сети невозможно сделать однозначный выбор. Одним из решений, позволяющим улучшить точность распознавания, может стать применение нескольких каналов при кодировании трассы в форме изображения как при обучении, так и при использовании модели. Предполагается, что, расширив способ представления трассы в форме изображения, модель сможет определять более сложные зависимости. Кроме того, использование *multilabel классификации*, варианта классификации, когда каждому объекту может соответствовать

несколько классов, может помочь избежать проблемы наслоения в исходных данных.

## Заключение

В данной работе проанализированы подходы по использованию методов машинного обучения в process mining применительно к задаче выделения паттернов поведения. Показано, что применение статистических методов реализуемо к исходным журналам событий, то есть минуя этап синтеза модели. Представлена реализация метода выделения поведенческих паттернов из определённого набора в журналах событий при помощи Image data engineering и свёрточных нейронных сетей. Ожидается, что, расширив способ представления трассы и использовав multilabel классификацию, удастся получить более точные результаты. Одно из направлений дальнейшей работы посвящено данному вопросу. К еще одному направлению работы можно отнести исследование по альтернативным представлениям журнала событий в виде матрицы и возможности обучения нейронной сети на этих данных. Планируется разработать способ представления журнала событий целиком в виде единственного изображения, после чего применить методы определения объектов (*objects / patterns detection*) на изображении для более точного обнаружения сразу нескольких паттернов.

## Список использованных источников


1. Aalst W. Van Der *Process Mining. Data Science in Action*. - 2-е изд. - Berlin: Springer-Verlag, 2016. - 486 с.
2. Aalst W. Van Der, A.J.M.M. Weijters, L. Maruster *Workflow Mining: Discovering Process Models from Event Logs* // IEEE Transactions on Knowledge and Data Engineering. - IEEE, 2004. - С. 1128 - 1142.
3. Sommers D., Menkovski V., Fahland D. *Process Discovery Using Graph Neural Networks* // 2021 3rd International Conference on Process Mining. - IEEE , 2021. - С. 40 - 47.
4. Pasquadibisceglie V., Appice A., Castellano G., Malerba D. *Using Convolutional Neural Networks for Predictive Process Analytics* // 2019 International Conference on Process Mining. - IEEE, 2019. - С. 129 - 136.
5. Ekene Obodoekwe, Xianwen Fang, Ke Lu *Convolutional Neural Networks in Process Mining and Data Analytics for Prediction Accuracy* // Electronics. - 2022 – URL: https://www.mdpi.com/2079-9292/11/14/2128 (дата обращения: 04.01.2023)
6. Gacitua-Decar V., Pahl C. *Automatic Business Process Pattern Matching for Enterprise Services Design* // 2009 World Conference on Services - II. - IEEE, 2009. - С. 111 - 118.

7. Shunin T., Zubkova N., Shershakov S. *Neural Approach to the Discovery Problem in Process Mining* // Analysis of Images, Social Networks and Texts. - Springer, 2019. - C. 261-273.
8. Nolle T., Seelige A., Thom N., Mühlhäuse M. *DeepAlign: Alignment-based Process Anomaly Correction Using Recurrent Neural Networks* // Advanced Information Systems Engineering. - Springer, 2020. - C. 319-333.
9. *Сверточные нейронные сети* // Викиконспекты Университета ИТМО URL: https://neerc.ifmo.ru/wiki (дата обращения: 07.05.2023).
10. Ramezani E., Fahland D., Van Dongen B., Aalst W. Van Der *Diagnostic Information in Temporal Compliance Checking* // Advanced Information Systems Engineering - 25th International Conference. - Springer, 2013. - C. 304-320.
11. Fukushima K. Cognitron: A self-organizing multilayered neural network // Biological Cybernetics. - Biological Cybernetics: Springer, 1975. - C. 121 — 136.
12. Bridle J. B. Training Stochastic Model Recognition Algorithms as Networks can Lead to Maximum Mutual Information Estimation of Parameters // NIPS'89: Proceedings of the 2nd International Conference on Neural Information Processing Systems. - Cambridge: MIT Press, 1989. - C. 211–217.
13. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton ImageNet Classification with Deep Convolutional Neural Networks // NeurIPS. – 2012
14. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition // arXiv preprint arXiv:1409.1556, 2014
15. Karim F., Majumdar S., Darabi H. CNNs for Time Series Classification // Journal of Computational Science, 2018
16. Kalchbrenner N., Grefenstette E., Blunsom P. A Convolutional Neural Network for Modelling Sentences // Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. 2014